

J3Script 0.9.1 - Complete Update Cheat Sheet

Precise new grammar with color-coded and bold J3Script syntax

J3Script 0.9.1-alpha | Compiler-sourced release supplement | 13 October 2026

This reference documents the 0.9.1 additions, together with shared commands and the seven exact language headers. The 0.9 baseline quick reference is appended after these pages. Existing source syntax remains valid unless explicitly superseded here.

Language selection (exact spelling)

Language	Generated output	Notes
HTML	.html	Document grammar
CSS	.css	Stylesheet grammar
JavaScript	.js	Standalone JS
C	.c	Native compile optional
SQL	.sql	MySQL/MariaDB
Python	.py	Standalone by default
PHP	.php	Standalone by default

Classes, fields, methods and construction

Form	Meaning
class NAME	Declare standalone class, top-level only
field NAME default EXPR	Declare instance field; evaluated by constructor
to METHOD ARG, ARG	Declare instance method; self is implicit
set self.FIELD to EXPR	Mutate a declared instance field
set VAR to CLASS()	Construct a class instance
VAR.METHOD(ARG, ARG)	Call statically resolved method
VAR.FIELD	Read instance field

```
PHP Counter
class Counter
  field value default 0
  to add amount
    set self.value to self.value + amount
    give back self.value
set current to Counter()
print current.add(3)
```

Classes are currently supported by Python console, JavaScript, PHP standalone, and C standalone, but not database-app DSLs. Constructors have no arguments. Method calls require a statically inferable receiver. No class inheritance, interfaces, or access modifiers.

Asynchronous task handles

Form	Purpose / targets
start FUNCTION ARG, ARG into HANDLE	Launch Python thread-pool or JavaScript Promise work
await HANDLE into RESULT	Wait for completed task and assign its result
C / PHP	Compilation rejects start/await (not implemented)

```
Python Async Demo
to double x
  give back x * 2
start double 8 into work
await work into value
print value
```

Task start calls user functions, not built-ins. Cancellation, portable synchronization, and C/PHP async are not part of 0.9.1.

Local package lock command

```
python -m j3c lock helpers@1.0.0 helpers.j3 --project .
```

Packages use a schema-1 j3-packages.json manifest with exact version, project-local source path and SHA-256. Import using **use package "helpers@1.0.0"**. No registry or transitive resolver.

Core functions and loops (unchanged)

Syntax	Meaning
to NAME PARAM, PARAM	Define a function
give back EXPR	Return from function
do NAME ARG, ARG into RESULT	Statement function call
NAME(ARG, ARG)	Expression function call in standalone targets
if EXPR / otherwise if / otherwise	Conditional
repeat N times	Counted loop
repeat while EXPR	Conditional loop
for each ITEM in EXPR	Collection loop
stop / skip	Break / continue
set NAME to EXPR	Assign variable
print EXPR	Produce output
make list NAME	Empty list
make NAME with FIELD EXPR	Create a record with named fields

Built-in function families

Category	Names
Math	absolute, floor, ceiling, power, modulo, square_root, round_number, minimum, maximum, clamp, sum_numbers
Text	lowercase, uppercase, trim, length, split, join, replace, starts_with, ends_with, to_text
Data / types	at, contains, range, keys, list_slice, is_number, is_text, is_list, is_nothing, to_number
Python-console-only	json_parse, json_stringify, http_get, arguments

IDE shortcuts

Control	Action
Tab	Accept suggestion or indent
Ctrl+Space	Request autocomplete
Ctrl+Enter	Build all
Run JS	Execute compiled JavaScript in isolated worker (trusted code only)
Save source	Download source ZIP
Recovery	Browser-local IndexedDB autosave and restore

Correctness note: The J3Script compiler is the final authority. The official bold/color styling in the website and guide is a display rule only, not mandatory characters or formatting within .j3 source.

Language update & quick cheat sheet

These pages document actual changes from J3Script 0.8.0. The complete 0.8 syntax reference follows; its older version headings refer to the baseline, not a second compiler.

The seven real-language headers (unchanged)

HTML, CSS, JavaScript, C, SQL, Python, PHP. Each selects the named output language. Python/PHP remain standalone by default; [project app](#) and [project plugin](#) are explicit specialized output modes, not languages. No legacy headers are accepted.

Shared application expression calls (new)

The Python and PHP database-application VMs now support the 31 pure standalone standard-library functions, list indexing, and function calls nested inside expressions. Filesystem, HTTPS and CLI operations remain unavailable from the web application VM. Runtime execution is bounded.

```
Python Customers
  project app
  to double n
    give back n * 2
  page home
    title "Contacts"
    when button "Calculate" is clicked
      set n to double(sum_numbers([2, 4, 6]))
      show message "Answer: {n}"
```

Version-pinned local packages (new)

use package "helpers@1.0.0" imports only local project source. The exact package name, semantic version, local .j3 path and SHA-256 checksum must be recorded in j3-packages.json. The compiler rejects mismatches and project-root escapes. No remote package installs are performed.

```
Python
  use package "helpers@1.0.0"
  print triple(7)

{
  "schema": 1,
  "packages": {"helpers": {
    "version": "1.0.0", "path": "helpers.j3",
    "sha256": "REPLACE_WITH_64_CHARACTER_HASH"
  }}
}
```

Web Studio local autosave (new)

The browser IDE now stores a project snapshot in IndexedDB on the device, prompts for recovery on reload, and displays a saving state. Source remains local to that browser. Browser quota, private browsing, eviction and user clearance can remove the saved copy: continue exporting source ZIP backups.

Quick syntax additions

Feature	Exact form	Scope
Nested function call	<code>double(sum_numbers([2,4]))</code>	App Python/PHP and standalone
Index value	<code>items[0]</code> or <code>items[-1]</code>	App Python/PHP; standalone
31 pure built-ins	<code>sum_numbers</code> , <code>clamp</code> , <code>keys</code> , etc.	Apps plus standalone
Local packages	use package "name@1.0.0"	Standalone JavaScript/C/Python/PHP modules
Package lock	<code>j3-packages.json</code>	Project source directory
Browser recovery	Automatic, on this device	Web Studio; not compiler syntax

Safety/portability notes

The four host-specific functions `json_parse`, `json_stringify`, `http_get` and `arguments` remain Python-console-only. Binary builds require local C tooling. WordPress is a deployment platform, never a J3Script language header.

J3SCRIPT / VERSION 0.8.0

J3Script Complete Cheat Sheet

Keywords, functions, selectors, statements, types and project modes.

Every documented command family for the seven current language headers, organized by context and color-coded to match Web Studio.

Bold blue keywords

Cyan actions

Purple types

Green strings

0.8.0 — ALL 35 currently implemented standalone functions

31 shared functions + 4 Python-console-only functions. The shared functions are validated across standalone Python, PHP, JavaScript and C; these are not automatically available in database-app button actions.

Function	Where	Purpose
absolute 1 arg	Python / PHP / JavaScript / C standalone	Absolute numeric magnitude
floor 1 arg	Python / PHP / JavaScript / C standalone	Round down to greatest integer not above value
ceiling 1 arg	Python / PHP / JavaScript / C standalone	Round up to least integer not below value
lowercase 1 arg	Python / PHP / JavaScript / C standalone	Convert text to lowercase
uppercase 1 arg	Python / PHP / JavaScript / C standalone	Convert text to uppercase
trim 1 arg	Python / PHP / JavaScript / C standalone	Remove surrounding whitespace from text
length 1 arg	Python / PHP / JavaScript / C standalone	Count list items or text characters
power 2 arg	Python / PHP / JavaScript / C standalone	Raise a number to a numeric power
modulo 2 arg	Python / PHP / JavaScript / C standalone	Remainder using floor-based division
at 2 arg	Python / PHP / JavaScript / C standalone	Read a list/text index or record field; negative indexes supported for lists
contains 2 arg	Python / PHP / JavaScript / C standalone	Check membership in a list or substring in text; record key membership
range 2 arg	Python / PHP / JavaScript / C standalone	Create half-open integer sequence from start to end, capped at 100,000 items
split 2 arg	Python / PHP / JavaScript / C standalone	Break text on a nonempty delimiter, producing a list
join 2 arg	Python / PHP / JavaScript / C standalone	Combine list values using a text separator
replace 3 arg	Python / PHP / JavaScript / C standalone	Replace all nonempty text matches with another string
to_text 1 arg	Python / PHP / JavaScript / C standalone	Convert supported J3Script value to its printable text form
to_number 1 arg	Python / PHP / JavaScript / C standalone	Parse finite numeric value from text; invalid text raises an error

Function	Where	Purpose
starts_with 2 arg	Python / PHP / JavaScript / C standalone	True if text begins with prefix (text, prefix)
ends_with 2 arg	Python / PHP / JavaScript / C standalone	True if text ends with suffix (text, suffix)
square_root 1 arg	Python / PHP / JavaScript / C standalone	Square root; rejects negative values (number)
round_number 1 arg	Python / PHP / JavaScript / C standalone	Nearest whole number; half away from zero (number)
minimum 2 arg	Python / PHP / JavaScript / C standalone	Smaller of two numeric values (a, b)
maximum 2 arg	Python / PHP / JavaScript / C standalone	Larger of two numeric values (a, b)
clamp 3 arg	Python / PHP / JavaScript / C standalone	Bound a value between minimum and maximum (n, low, high)
is_number 1 arg	Python / PHP / JavaScript / C standalone	Check that the value is numeric and not Boolean
is_text 1 arg	Python / PHP / JavaScript / C standalone	Check whether value is text
is_list 1 arg	Python / PHP / JavaScript / C standalone	Check whether value is a list
is_nothing 1 arg	Python / PHP / JavaScript / C standalone	Check whether value is nothing/null
keys 1 arg	Python / PHP / JavaScript / C standalone	Return ordered field names from a record
list_slice 3 arg	Python / PHP / JavaScript / C standalone	Return items from start up to but not including end; negatives allowed
sum_numbers 1 arg	Python / PHP / JavaScript / C standalone	Sum numeric items of a list; rejects nonnumeric entries
json_parse 1 arg	Python console only	Parse JSON text into lists/records/numbers/booleans (Python console only)
json_stringify 1 arg	Python console only	Encode a JavaScript value as JSON (Python console only)
http_get 1 arg	Python console only	Retrieve up to 1 MB of HTTPS text; 5 second timeout (Python console only)
arguments 0 arg	Python console only	Return CLI arguments as a list, excluding script path (Python console only)

0.8 expressions and example

```
JavaScript
set numbers to [2, 4, 6]
set total to sum_numbers(numbers)
print "Total: {total}"
print round_number(-2.5)
print starts_with("J3Script", "J3")
```

All function calls can be used in expressions, e.g. `set answer to clamp(42, 0, 10)`, where the standalone grammar supports it. The official color scheme is unchanged: **bold blue keywords**, **bold cyan actions**, **purple types**, **green strings**, muted gray comments and ordinary identifiers.

0.8 limitations you should not overlook

Standalone Python/PHP selection does not imply an application. `project app` must be explicit. Only Python console allows JSON, CLI arguments and HTTPS builtins; C, PHP and JavaScript compilation rejects them. Apps use a restricted VM and do not yet expose the full 31-function core. There is no WordPress programming-language header.

J3Script 0.8.0 — Quick Reference

Use this section at your keyboard. Color legend: **bold blue = keyword / syntax**; **bold cyan = action/event**; **purple = type**; **green = string**; **gray = comment**; **white = user-defined name**. Bold/color are **editor and documentation formatting only**, never mandatory characters in a `.j3` file.

A. All legal language headers and project declarations

Header / project line	Valid purpose	Example
HTML (optional title)	HTML page	HTML or HTML "Welcome"
CSS	CSS file	CSS
JavaScript NAME	JavaScript program / DOM logic	JavaScript My Events
C NAME	C source + optional native binary	C Number Helper
SQL [DATABASE]	SQL DDL/DML generator	SQL realestate
Python (optional name)	Standalone Python <code>.py</code> script by default	Python
Python NAME + project app	Explicit Python web app	Python My CRM then project app
Python + project console	Optional explicit console mode	project console
Python NAME + project module	Python helper functions	project module
PHP (optional name)	Standalone PHP <code>.php</code> script by default	PHP
PHP NAME + project app	Standalone PHP web app	PHP My CRM then project app
PHP NAME + project wordpress-app	PHP app for WordPress	project wordpress-app

Header / project line	Valid purpose	Example
PHP NAME + <code>project plugin</code>	Traditional PHP plugin	<code>project plugin</code>
PHP NAME + <code>project module</code>	PHP plugin helper functions	<code>project module</code>
JavaScript/C NAME + <code>project module</code>	Shared logic module	<code>project module</code>

Rejected top-level language words: `page`, `style`, `script`, `program`, `data`, `app`, `plugin`, `WordPress`, `WordPress Plugin`. These may still occur **inside legal source** where the grammar defines them, especially `page`, `style`, and `script` in app/HTML internals.

B. Core logic: every common command family

Syntax	Meaning	Context
<code>set</code> NAME <code>to</code> EXPR	Assign a variable	Shared standalone / app button logic
<code>add</code> EXPR <code>to</code> NAME	Numeric add or list append	Shared logic
<code>subtract</code> EXPR <code>from</code> NAME	Subtract and store	Shared logic
<code>multiply</code> NAME <code>by</code> EXPR	Multiply and store	Shared logic
<code>divide</code> NAME <code>by</code> EXPR	Divide and store	Shared logic
<code>print</code> EXPR	Print/log/emit depending on runtime	Logic; in app shows message
<code>make list</code> NAME	Empty list	Logic
<code>make</code> NAME <code>with</code> FIELD EXPR, ...	Record	Logic
<code>if</code> CONDITION	True branch	Logic
<code>otherwise if</code> CONDITION	Else-if branch	Logic
<code>otherwise</code>	Default branch	Logic
<code>repeat</code> N times	Counted loop	Logic
<code>repeat while</code> CONDITION	Conditional loop	Logic
<code>for each</code> ITEM <code>in</code> EXPR	Iterate collection	Logic
<code>stop</code>	Break nearest loop	Inside loop
<code>skip</code>	Continue next iteration	Inside loop
<code>to</code> FUNCTION ARG, ARG	Declare a function	Top-level logic / app-level
<code>give back</code> EXPR	Return function result	Function
<code>do</code> FUNCTION ARG, ARG <code>into</code> NAME	Call and capture result	Logic
FUNCTION(ARG, ...)	Expression call	0.8.0 standalone logic only
<code>task</code> NAME, LABEL	Protected operation block	Logic
<code>fail</code> KIND "MESSAGE"	Raise named error	Logic
<code>catch</code> KIND <code>as</code> VARIABLE	Handle named error	Within protected block

Syntax	Meaning	Context
<code>catch any as VARIABLE</code>	Handle supported recoverable errors	Within protected block
<code>use module "FILE.j3"</code>	Import compiled helper functions	Standalone JS/C/Python console/PHP plugin
<code>rules</code>	Rule-selector container	Standalone logic / app
<code>use rules "FILE.rules"</code>	Import shared execution rules	Supported logic targets

C. All expression, value, collection and comparison forms

Token / form	Meaning / example
<code>"TEXT"</code>	Quoted text: <code>"Hello"</code>
<code>yes, no</code>	Boolean values
<code>nothing</code>	Missing/empty value marker
<code>12, -3, 2.5</code>	Numeric literals
<code>[A, B, C]</code>	List literal
<code>+, -, *, /</code>	Arithmetic operators; spaces around subtraction recommended
<code>=, !=, <, <=, >, >=</code>	Comparisons (= is equality, not assignment)
<code>and, or, not</code>	Boolean operations
<code>(EXPR)</code>	Expression grouping
<code>"Hello {name}"</code>	Text interpolation in supported logic contexts
<code>NAME.field</code>	Record field read
<code>LIST[0], LIST[-1]</code>	Indexed access in standalone 0.8.0 logic
<code>RECORD["field"]</code>	Indexed record field in standalone 0.8.0 logic
<code>LIST.size</code>	Number of list items
<code>LIST.first, LIST.last</code>	First/last list member
<code>FUNCTION(a, b)</code>	Function call inside an expression (standalone 0.7)
<code>-- COMMENT</code>	Comment outside quoted string

D. Full standard library: all 17 names

Function	Purpose	Example
<code>absolute(x)</code>	Positive magnitude	<code>absolute(-5)</code>
<code>floor(x)</code>	Round down	<code>floor(2.9)</code>
<code>ceiling(x)</code>	Round up	<code>ceiling(2.1)</code>
<code>power(x, y)</code>	Exponent	<code>power(2, 3)</code>
<code>modulo(x, y)</code>	Remainder	<code>modulo(11, 3)</code>

Function	Purpose	Example
<code>lowercase(text)</code>	Lowercase	<code>lowercase("HI")</code>
<code>uppercase(text)</code>	Uppercase	<code>uppercase("hi")</code>
<code>trim(text)</code>	Trim edges	<code>trim(" hi ")</code>
<code>length(value)</code>	Count items/characters	<code>length("abc")</code>
<code>at(value, index)</code>	Index/key lookup	<code>at([4, 9], 0)</code>
<code>contains(collection, value)</code>	Membership test	<code>contains([1, 2], 2)</code>
<code>range(start, end)</code>	Integers, end excluded	<code>range(1, 4)</code>
<code>split(text, delimiter)</code>	Split into list	<code>split("a,b", ",")</code>
<code>join(list, delimiter)</code>	Concatenate with separator	<code>join(["a", "b"], "-")</code>
<code>replace(text, find, with)</code>	Text replacement	<code>replace("a-b", "-", " ")</code>
<code>to_text(value)</code>	Convert to text	<code>to_text(10)</code>
<code>to_number(value)</code>	Parse numeric value	<code>to_number("10")</code>

Availability: standalone JavaScript, C, Python console, and traditional PHP plugin runtime. **Do not assume** they work in app button logic. `range` caps at 100,000 items, and 0.8.0 runtime edge cases can vary by backend.

E. Standalone tasks and rule selectors

Syntax or selector	What it matches / does
<code>task: on error log</code>	Every task block
<code>.safe: on error ignore</code>	Blocks labeled <code>safe</code>
<code>#payroll: on error retry 3</code>	Named block <code>payroll</code>
<code>task.safe: on error log</code>	Tasks with label <code>safe</code>
<code>task #specific</code>	Descendant named block in matching hierarchy
<code>on error pass up</code>	Propagate failure
<code>on error stop</code>	Stop current block
<code>on error log</code>	Log failure
<code>on error ignore</code>	Ignore recoverable failure
<code>on error retry N</code>	Bounded retry where supported; 0–5 in apps
<code>run in order</code>	Sequential execution
<code>run in background</code>	Only supported in certain standalone targets
<code>timeout N seconds</code>	Execution policy; enforcement depends on runtime
<code>memory free at end</code>	Release/end-of-block policy

Syntax or selector	What it matches / does
<code>memory keep</code>	Accepted in some legacy runtimes; not Python console
<code>log on</code> , <code>log off</code>	Toggle block execution logging

F. HTML header: element and document inventory

Category	All recognized words / syntax
Document directives	<code>description "..."</code> ; <code>use style FILE</code> ; <code>use script FILE</code>
Semantic containers	<code>header</code> , <code>footer</code> , <code>nav</code> , <code>section</code> , <code>main</code> , <code>article</code> , <code>aside</code>
Headings	<code>title</code> , <code>subtitle</code> , <code>heading</code> , <code>small-heading</code>
Text	<code>text</code> , <code>quote</code> , <code>bold</code> , <code>note</code>
Media	<code>image "path" "alt"</code> ; <code>video "path"</code>
Navigation	<code>link "label" to "url"</code>
Lists	<code>list</code> , <code>numbered</code> , <code>item</code>
Forms	<code>form</code> , <code>input</code> , <code>area</code> , <code>label</code> , <code>button</code>
Tables/rules	<code>table</code> , <code>row-of</code> , <code>cell</code> , <code>line</code>
Input types	<code>email</code> , <code>password</code> , <code>number</code> , <code>tel</code> , <code>date</code> , <code>url</code> , <code>search</code> , <code>color</code> , <code>file</code>
Input/button options	<code>required</code> , <code>submit</code>
IDs and classes	<code>section id-name</code> ; <code>card</code> , <code>featured</code>
Other element kind	Becomes a <code><div class="kind"></code> container

G. CSS header: every recognized selector/state/shortcut

Category	Recognized syntax / behavior
Elements	<code>page</code> (body), <code>title</code> (h1), etc., from HTML kind mapping
Identifier selectors	<code>#element-id</code> , <code>.class-name</code> , <code>card.featured</code>
Descendant selection	<code>nav link</code> , <code>card image</code>
States	<code>when hovered</code> , <code>when focused</code> , <code>when pressed</code> , <code>when first</code> , <code>when last</code>
Breakpoints	<code>on phone</code> ≤600px; <code>on tablet</code> ≤900px; <code>on desktop</code> ≥901px
Layout	<code>layout row</code> , <code>layout column</code> , <code>layout grid N</code> , <code>layout center</code>
Typography	<code>size</code> , <code>font</code> , <code>align</code> , <code>bold</code> , <code>italic</code> , <code>uppercase</code> , <code>underline</code> , <code>center</code>
Box	<code>padding</code> , <code>margin</code> , <code>width</code> , <code>height</code> , <code>gap</code> , <code>radius</code> , <code>rounded</code> , <code>border</code>
Other shortcuts	<code>shadow on/none</code> , <code>hidden</code> , <code>pointer</code>
Multiword properties	max <code>width</code> , min <code>width</code> , max <code>height</code> , min <code>height</code> , <code>line height</code> , letter spacing

Category	Recognized syntax / behavior
Pass-through examples	<code>color</code> , <code>background</code> , <code>opacity</code> , <code>cursor</code> , <code>display</code> , <code>position</code> , <code>overflow</code> , <code>transition</code> , <code>transform</code> , <code>z-index</code> , <code>content</code>
Syntax layout	selector: property value, property value OR indented properties

Other CSS properties may be passed through after spelling conversion, but JScript does **not** provide a full CSS grammar/validator. Color values like `#245bd7` are ordinary CSS values, not JScript reserved words.

H. JavaScript header: browser/logic inventory

Syntax	Effect
<code>when #id is clicked</code>	<code>click</code> event
<code>when #id is changed</code>	<code>change</code> event
<code>when #id is submitted</code>	<code>submit</code> event
<code>when #id is hovered</code>	<code>mouseenter</code> event
<code>when #id is typed in</code>	<code>input</code> event
<code>when page</code> loads	Browser page-load handler
<code>read #id into VAR</code>	Read control value
<code>put EXPR into #id</code>	Set element content/value as generated
<code>show #id / hide #id</code>	Change visibility
<code>open FILE as VAR</code>	Existing standalone file operation, Node not browser
<code>make channel</code> , <code>send</code> , <code>receive</code>	Legacy concurrency operations, target-specific

I. C header: native compilation inventory

Syntax / requirement	Meaning
<code>C</code> Program Name	Generate C source and optional executable
<code>use c library "libm"</code>	Declare linked native library
<code>to pow a as number, b as number</code> gives <code>number</code>	Native signature under C library
<code>number</code> , <code>int32</code> , <code>byte</code> , <code>text</code> , <code>nothing</code>	C foreign-function signature types
<code>make channel NAME / send / receive</code>	Supported legacy runtime operations; semantics differ from JS
<code>free NAME</code>	Resource free in supported standalone logic
<code>--cc PATH</code>	Select C compiler
<code>--run</code>	Compile/run local binary, if toolchain available

J. SQL header: table types, modifiers and query verbs

Category	Complete recognized forms
Header	<code>SQL</code> or <code>SQL DATABASE</code>
Define table	<code>table</code> NAME (indented column declarations)
Types	<code>number</code> , <code>big number</code> , <code>decimal</code> , <code>money</code> , <code>text</code> , <code>text N</code> , <code>long text</code> , <code>yes/no</code> , <code>date</code> , <code>time</code> , <code>email</code> , <code>json</code> , <code>link to</code> TABLE, <code>choice</code> of "A" or "B"
Modifiers	<code>key</code> , <code>auto</code> , <code>required</code> , <code>unique</code> , <code>default</code> EXPR, <code>now</code>
Insert	<code>add to</code> TABLE + indented COL VALUE, ... rows
Read	<code>find</code> TABLE where CONDITION
Project columns	<code>find</code> COL and COL <code>from</code> TABLE
Sort	<code>sorted by</code> FIELD / <code>sorted by</code> FIELD <code>descending</code>
Limit	<code>limit</code> N
Update	<code>change</code> TABLE where CONDITION + assignments
Delete	<code>remove from</code> TABLE where CONDITION
Conditions	Comparisons, and, or, not, SQL-valid literals
Generated dialect	MySQL/MariaDB, InnoDB, utf8mb4

K. Python console header: complete I/O inventory

Syntax	Action
<code>project console</code>	Output independent <code>.py</code> file
<code>ask</code> PROMPT <code>into</code> VAR	Terminal input as text
<code>write</code> TEXT <code>to file</code> PATH	Overwrite UTF-8 file
<code>read text from</code> PATH <code>into</code> VAR	Read UTF-8 file
<code>open</code> "FILE.csv" <code>as</code> ROWS	Read supported file; CSV may produce records
<code>use module</code> "HELPERS.j3"	Import J3Script function module
<code>project module</code>	Declare Python helper module
<code>print</code> EXPR	Console output
<code>--run</code>	Run exported Python console program locally
Execution guards	1,000,000 steps, recursion limit 64

L. App grammar: declarations, fields, pages, actions, filters

Category	Complete major constructs
Project	<code>Python</code> NAME / <code>PHP</code> NAME plus explicit <code>project app</code> , or PHP <code>project wordpress-app</code>
App root	<code>accounts</code> , <code>database</code> , <code>page</code> , <code>style</code> , <code>rules</code> , <code>use rules</code> , top-level <code>to</code> functions
Database field	<code>FIELD as TYPE</code> , <code>required</code> , <code>unique</code> , <code>default</code> VALUE
App types	<code>text</code> , <code>long text</code> , <code>email</code> , <code>number</code> , <code>decimal</code> , <code>money</code> , <code>yes/no</code> , <code>date</code> , <code>time</code> , <code>phone</code> , <code>url</code> , <code>choice</code> of ..., <code>link to</code> TABLE
Seeds	<code>add</code> FIELD VALUE, FIELD VALUE inside database
Pages	<code>page</code> NAME; options <code>home</code> , <code>public</code> , <code>roles</code> owner and admin and editor and viewer
Text	<code>title</code> , <code>subtitle</code> , <code>heading</code> , <code>text</code> with quoted content
Link	<code>link "LABEL" to page</code> TARGET
Records	<code>show</code> TABLE <code>as table/list</code> , <code>show count</code> of TABLE
Columns	<code>with</code> FIELD and FIELD
Display options	<code>searchable</code> , <code>editable</code> , <code>removable</code> , <code>oldest first</code> , <code>limit</code> N, <code>page size</code> N
Filter	where FIELD OP LITERAL with and joins
Filter operators	<code>=</code> , <code>!=</code> , <code><</code> , <code><=</code> , <code>></code> , <code>>=</code> , <code>contains</code>
Form	<code>form</code> FORM-NAME with nested inputs and buttons
Input	<code>input</code> NAME, <code>label</code> "...", <code>placeholder</code> "...", <code>required</code>
Button	<code>when button "LABEL" is clicked</code>
Validation	<code>validate</code> FORM
Insert	<code>save</code> FORM <code>to</code> DATABASE
Update pending input	<code>put</code> EXPR <code>into input</code> FIELD
Flash	<code>show message</code> EXPR
Navigation	<code>open page</code> TARGET (final action)
Auth identity	<code>user.id</code> , <code>user.email</code> , <code>user.role</code>
App logic	<code>set</code> , <code>make</code> , arithmetic updates, <code>if</code> , loops, <code>do</code> , <code>give back</code> , <code>task</code> , <code>catch</code> , <code>fail</code> , <code>print</code>
App controls	Error policy, transactional rollback, 10k steps, 2s deadline, 32 nested calls
App unsupported	Standalone file I/O, background/channel/native ops, arbitrary class/network/package operations

M. PHP projects and WordPress-facing API

Setting or construct	Purpose
<code>PHP</code> NAME + <code>project app</code>	Standalone PHP app

Setting or construct	Purpose
<code>PHP</code> NAME + <code>project plugin</code>	Traditional WordPress plugin
<code>PHP</code> NAME + <code>project wordpress-app</code>	Complete application in WordPress
<code>PHP</code> NAME + <code>project module</code>	Traditional-plugin helper functions
<code>version</code> , <code>author</code> , <code>description</code>	PHP plugin metadata
<code>shortcode</code> NAME [PARAMS]	Register shortcode
<code>when</code> HOOK	Register action hook
<code>filter</code> HOOK PARAM	Register filter
<code>give back</code> EXPR	Return shortcode/filter result
<code>use module</code> "HELPERS.j3"	Import reusable PHP function source
<code>accounts</code> in standalone PHP app	Multi-user generated app
WordPress app roles	WordPress identity, capability mapping, nonce protection
Live requirement	PHP 8.1+; appropriate database driver; staging activation

N. CLI reference and IDE shortcuts

Command / control	Action
<code>python -m j3c check FILE</code>	Syntax/semantic check
<code>python -m j3c build FILE -o DIR</code>	Compile output
<code>python -m j3c build FILE --run</code>	Execute supported standalone targets
<code>python -m j3c serve APP.j3 --port 8000</code>	Serve app locally for development
<code>--backend python php wordpress all</code>	App export selection
<code>--cc /path/to/cc</code>	Native compiler path
<code>--host 127.0.0.1</code>	Bind locally; <code>0.0.0.0</code> exposes on network
IDE Tab	Accept suggestion or indent
IDE Enter	Accept suggestion / insert line and indent
IDE Ctrl+Space	Request autocomplete
IDE Escape	Close suggestion list
IDE Check all / Build all	Validate / generate outputs

Final caution: a cheat sheet is not a substitute for context

A spelling can mean a different thing depending on where it occurs: `page` is an internal app declaration, a CSS selector for the HTML body, or an English noun; `text` is an HTML element kind and an app/SQL type; `show` can update the DOM or display an app record list/message. J3Script's **language header + project mode +**

containing block determine the grammar. Bold/color are a presentation choice, not mandatory source markup.